Unityはじめるよ
～NavMesh基礎～

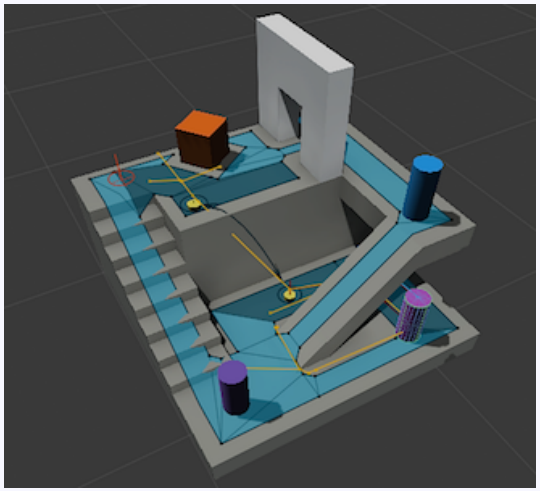
統合開発環境を内蔵したゲームエンジン
http://japan.unity3d.com/

※いろんな職業の方が見る資料なので説明を簡単にしてある部分があります。正確には本来の意味と違いますが上記理由のためです。ご了承ください。
この資料内の一部の画像、一部の文章はUnity公式サイトから引用しています。

NavMesh（ナビメッシュ）って？

NavMeshって？

ナビゲーションシステムによって、シーンのジオメトリから自動で生成されるナビゲーションメッシュを使用して、ゲーム世界の中を知的に動くキャラクターを作成することができます。
動的（Dynamic）障害物によってランタイムでのキャラクターのナビゲーションを変化させることができ、オフメッシュリンクのおかげで、扉の開きや出っ張りからの飛び降りなどの個別のアクションを作ることができます。っp



公式ドキュメントから引用
https://docs.unity3d.com/ja/current/Manual/Navigation.html

で、何ができるのさ？

- 自動で移動するNPCが作れるよ。
- マウスクリック地点にキャラを移動させることもできるよ。

障害物を自動で避けて、目的地まで移動可能！
使い方も非常に簡単！

イベントシーン等で、キャラクターを自動で移動させたい場合にも有効！

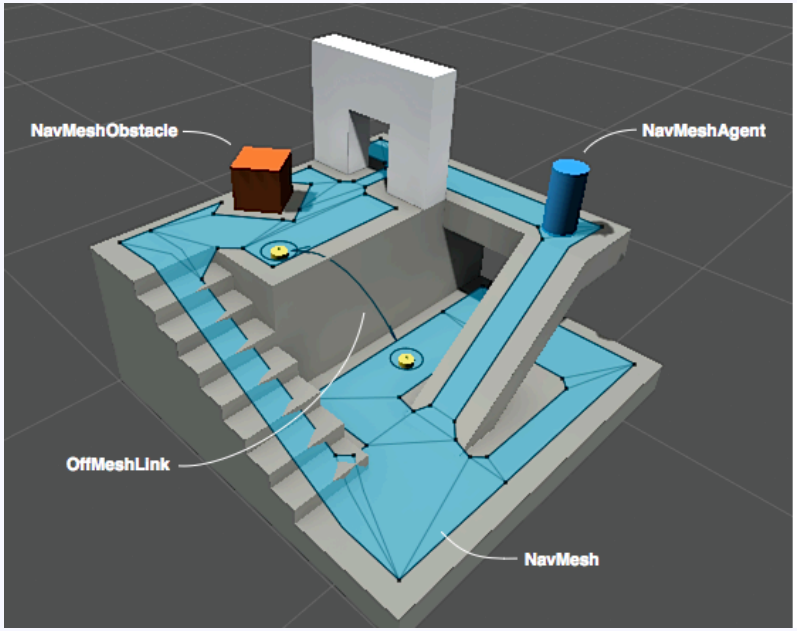
NavMeshの用語解説

ナビメッシュ（NavMesh）
（Navigation Mesh の略）は、ゲーム世界の中で歩行可能な面を描写するデータ構造で、ひとつの歩行可能な面から別の歩行可能な面へのパスを見つけることができるようにするものです。データ構造は、ステージのジオメトリ（形状）から自動で構築またはベイクされます。

ナビメッシュ エージェント（NavMesh Agent）
コンポーネントは、キャラクターがゴールに向かって進行する際に相互に回避し合うように作成する助けになります。エージェントはナビメッシュを使用してゲーム世界を認識（判断）し、互いに回避し合ったり、障害物を動かしたりする方法を理解しています。

オフメッシュリンク（Off-Mesh Link）
コンポーネントは、「歩行可能な面」を使って表現することのできないナビゲーション・ショートカットの組み込みを可能にします。例えば溝やフェンスを飛び越えたり扉を通り抜ける前にそれを開くことなどはすべてオフメッシュリンクとして描写できます。

ナビメッシュ障害物（NavMesh Obstacle）
コンポーネント。これによって、エージェントがゲーム世界をナビゲートする（動き回る）際に避けるべき障害物を、動かすことができます。例えば、物理システムによって制御された樽やクレートなども「障害物」の一例です。障害物が動いている間は、エージェントはそれを回避しようと最善を尽くします。障害物の動きが止まるとそれによってナビメッシュに穴が開けられるので、エージェントはそれを避けて通るために進行コースを変更することができます。あるいは、その静止した障害物が通り道を塞いでいる場合は、エージェントはよりよいルートを見つけることができます。



公式ドキュメントから引用

https://docs.unity3d.com/ja/current/Manual/nav-NavigationSystem.html

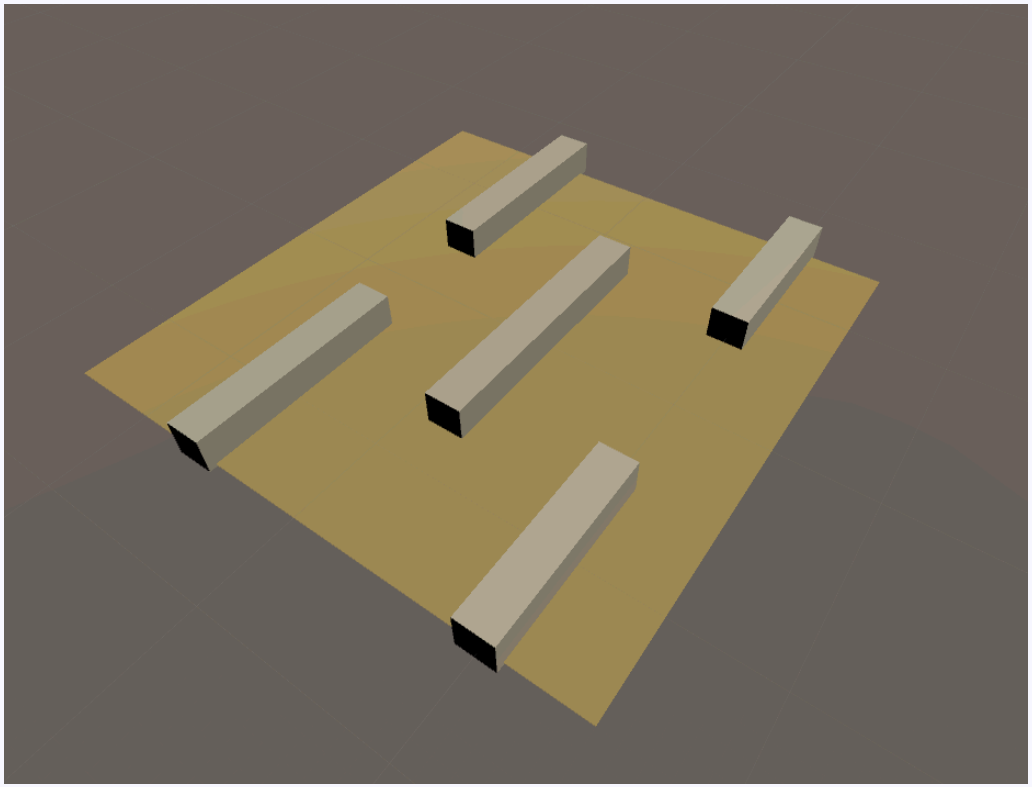
NavMeshの使い方

今回は、プレイヤーにNPCとしての犬が寄ってくる状況をNavMeshを使って再現するぜ。

1. 地形と障害物を作る
2. NavMeshの設定
3. NPCの作成(NavMeshAgent設定)
4. スクリプトで目的地を設定

１．地形と障害物を作る

地形と障害物を作る

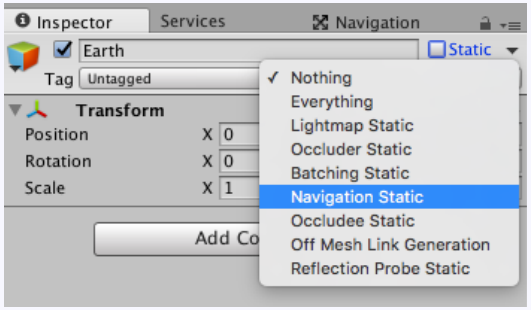


今回はシンプルに、
地面(Plane)と障害物(Cube)を置いただけ。

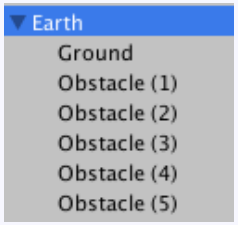
2．NavMeshの設定

NavMeshの設定

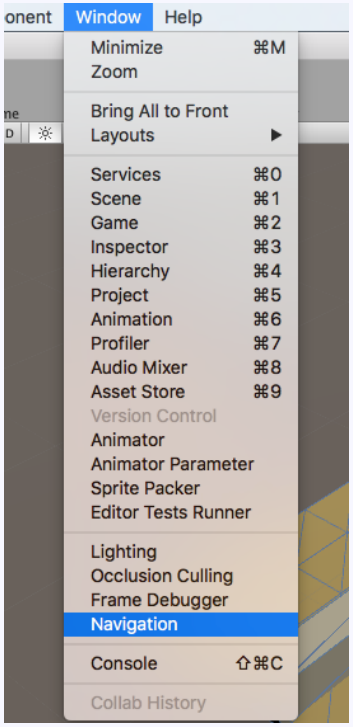
1．地形と障害物を**Navigation Static**にする。



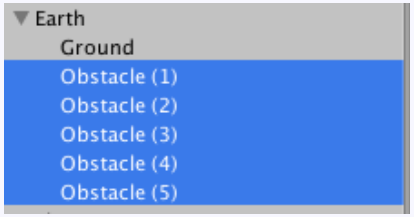
ちなみに地形と障害物はこんな感じで作ってある



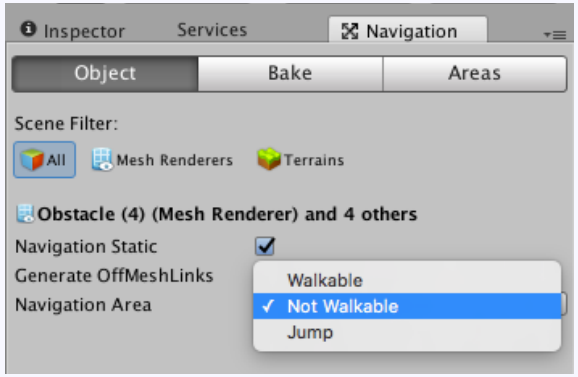
2．**メニュー > Window > Navigation**
でNavigationウィンドウを開く



NavMeshの設定



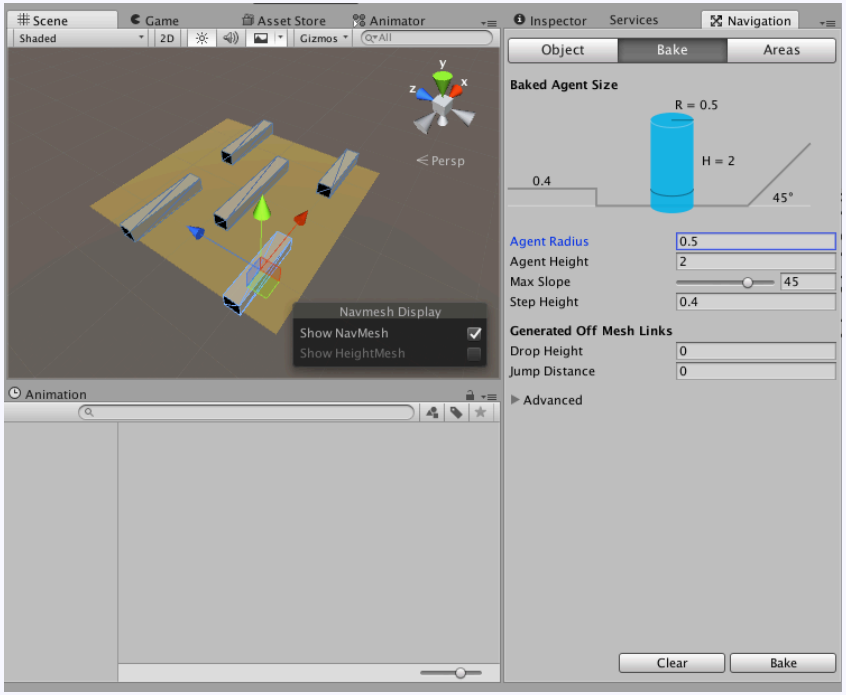
ヒエラルキービューで
障害物GameObjectを選択

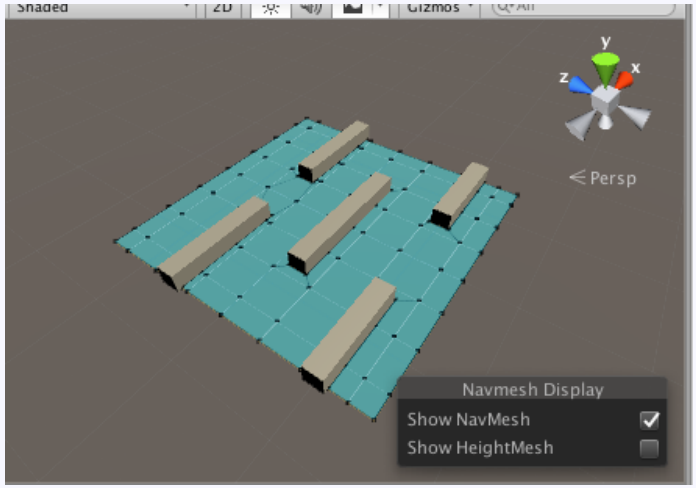


NavigationAreaをNot Walkableに

3．今回は障害物の上を歩かせたくないので、
ヒエラルキービューで障害物を選択し、
NavigationウィンドウのObjectタブ内の
NavigationAreaを**Not Walkable**にする。

NavMeshの設定

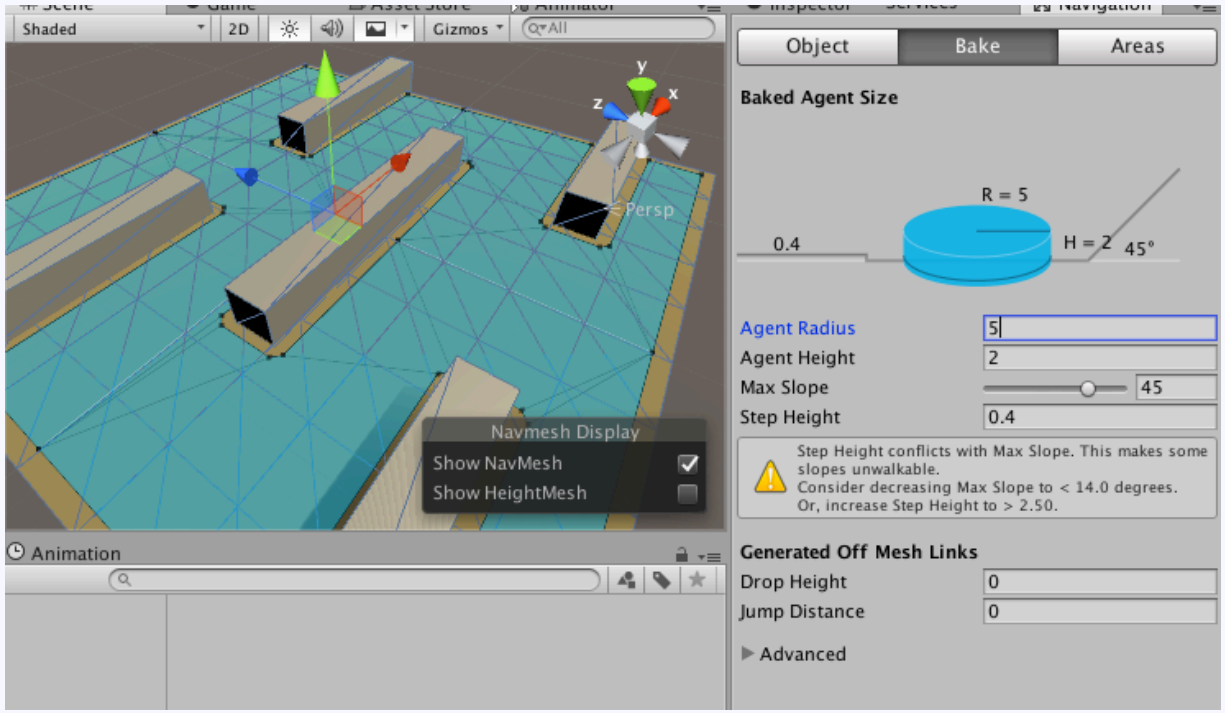




歩ける範囲が青い色で表示される

4．Bakeタブを開き**Bakeボタン**を押す。
そうすると歩ける範囲を自動で計算して表示してくれる。

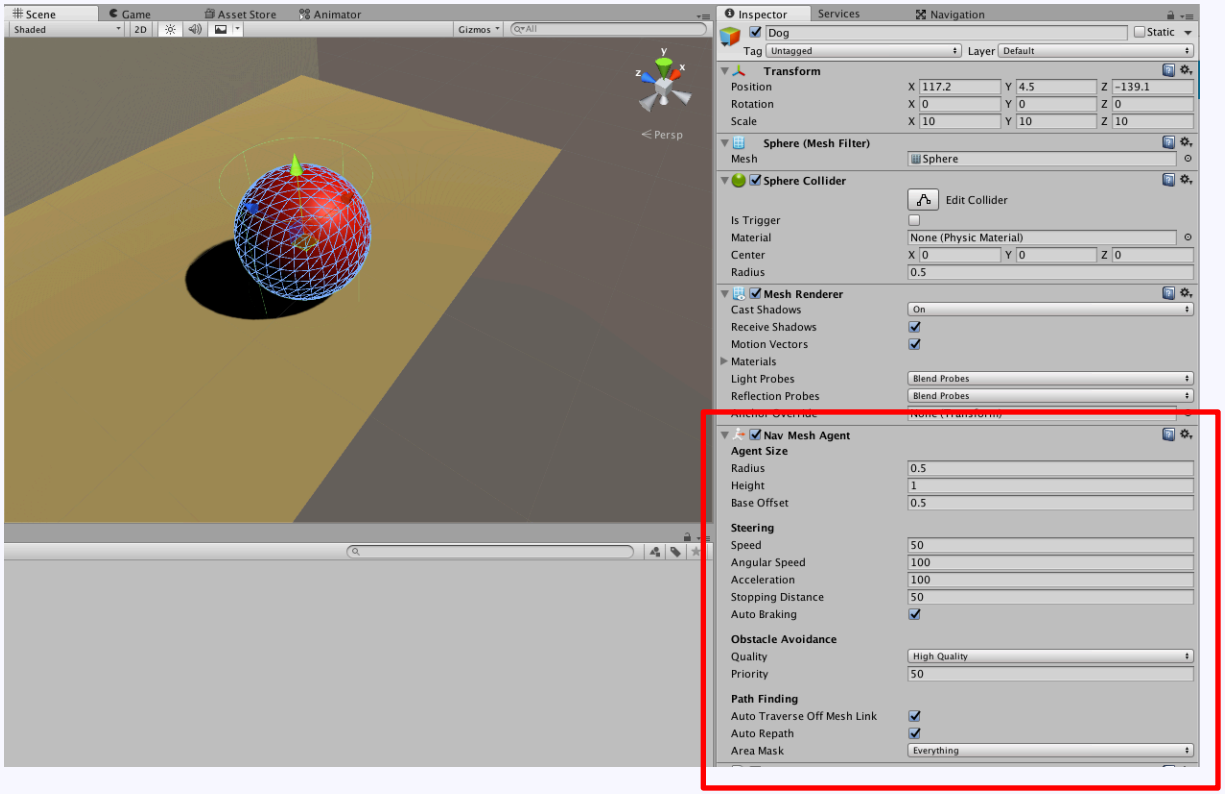
NavMeshの設定



5．歩ける範囲が障害物にぴったりくっつきすぎているので、
Agent Radius(キャラクターの半径)を変更し、再度Bake。

３．NPCの作成(NavMeshAgent設定)

NPCの作成(NavMeshAgent設定)



1．NPCとして「Dog」ゲームオブジェクトを作成し、**NavMeshAgentコンポーネント**を追加する。

AddComponent > Navigation > Nav Mesh Agent

NavMeshAgentの設定項目

プロパティー	説明
Agent Size	
Radius	エージェントの半径は、障害物や他のエージェント間の衝突の計算に使用されます。
Height	頭上にある障害物の下を通り抜けるのにエージェントが必要とする高さ。
Base offset	トランスフォーム ピボット ポイントに対するコリジョン シリンダー（円筒）のオフセット。
Steering	
Speed	動きの最高速度（単位： メートル／秒）。
Angular Speed	回転の最高速度（単位： 度／秒）
Acceleration	最高加速度（単位： メートル／秒の平方）
Stopping distance	ゴール位置に、この設定距離まで近づいたらエージェントが停まります。
Auto Braking	有効になっていると、目的地点の到着時にエージェントの移動速度が減速します。巡回のような、エージェントが複数のポイント間をスムーズに移動する必要がある場合には無効にする必要があります。
Obstacle Avoidance （障害物回避）	
Quality	障害物回避の品質。エージェントの数が多い場合には障害物回避の品質を落とすことで CPU 時間を節約することができます。回避を None に設定するとエージェント同士が衝突しなくなりますが、他のエージェントが障害物を積極的に回避するようになる訳ではありません。
Priority	エージェントは、回避においてこの数値より低いエージェントを無視するようになります。値は 0〜99 の間で設定し、数字が小さいほど優先度が高いことを意味します。
Path Finding （経路の探索）	
Auto Traverse OffMesh Link	オフメッシュリンクを自動で越えるようにしたい場合はオンにしてください。オフメッシュリンクを越えるのにアニメーションなどの特定の方法を使用したい場合はオフにしてください。
Auto Repath	有効になっていると、エージェントは部分的な経路の終了地点へ到達した際に、再度経路の探索を行います。目的地への経路がない場合は、目的地にもっとも近い到達可能場所への部分的な経路が作成されます。
Area Mask	Area Mask は、エージェントが経路を探索する際にどのナビゲーションエリアとコスト を確認するかを設定します。ナビメッシュのベーキング用にメッシュを準備する際に、各メッシュにエリアタイプを設定できます。例えば、階段を特殊なエリアタイプでマスクし、一部のキャラクターはその階段を使えないようにしたりすることが可能です。

公式ドキュメントから引用

https://docs.unity3d.com/ja/current/Manual/class-NavMeshAgent.html

４．スクリプトで目的地を設定

スクリプトで目的地を設定

1．NPCを動かすDogControllerスクリプトを作成 ゲームオブジェクトにアタッチしといてね。

```
using UnityEngine;
using System.Collections;

public class DogController : MonoBehaviour {

    public GameObject    TargetObject;       /// 目標位置

    NavMeshAgent        m_navMeshAgent;       /// NavMeshAgent


    // Use this for initialization
    void Start () {

        // NavMeshAgentコンポーネントを取得
        m_navMeshAgent = GetComponent<NavMeshAgent>();

    }

    // Update is called once per frame
    void Update () {

        // NavMeshが準備できているなら
        if(m_navMeshAgent.pathStatus != NavMeshPathStatus.PathInvalid) {

            // NavMeshAgentに目的地をセット
            m_navMeshAgent.SetDestination(TargetObject.transform.position);

        }
    }
}
```

目標GameObjectは Editorから指定する

コンポーネントを取得

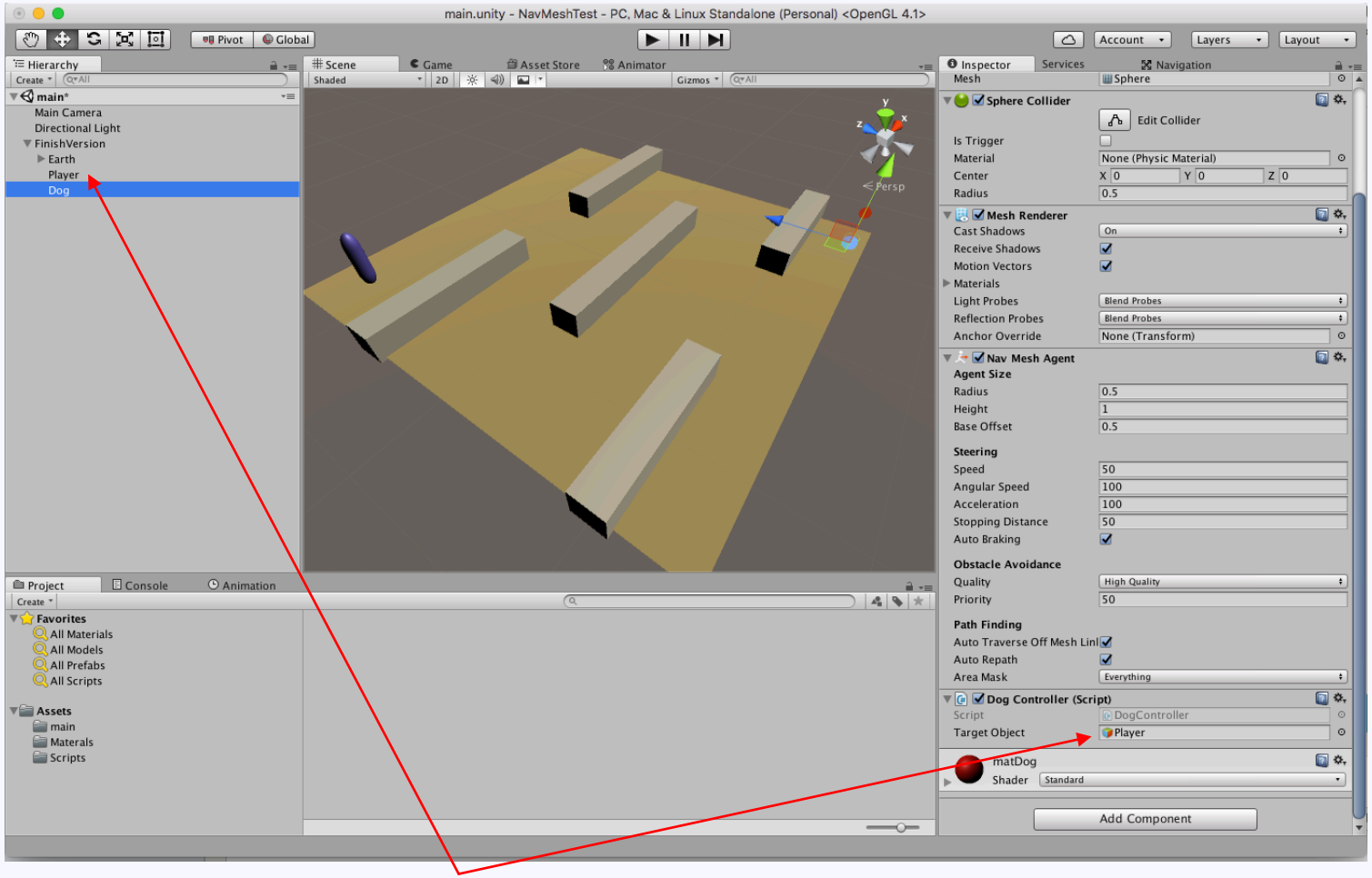
NavMeshAgentに目的地を設定 処理は重めなので、なるべくならUpdateに入れないほうがいい

スクリプトで目的地を設定

2．ついでにプレイヤーを動かすPlayerControllerスクリプトを作成 ゲームオブジェクトを作ってアタッチしといてね。

```
using UnityEngine;
using System.Collections;

public class PlayerController : MonoBehaviour {

    public float Speed = 1.0f;       // 移動速度 m/s

    // Update is called once per frame
    void Update () {
        Vector3 pos = transform.position;

        float moveLength = Speed * Time.smoothDeltaTime;

        if (Input.GetKey (KeyCode.UpArrow))
        {
            pos.z += moveLength;
        }
        else if (Input.GetKey (KeyCode.DownArrow))
        {
            pos.z -= moveLength;
        }
        else if (Input.GetKey (KeyCode.LeftArrow))
        {
            pos.x -= moveLength;
        }
        else if (Input.GetKey (KeyCode.RightArrow))
        {
            pos.x += moveLength;
        }

        transform.position = pos;
    }
}
```

3．DogControllerにPlayerゲームオブジェクトをセットしとく



完成
実行してみよう！

■まとめ

NavMeshを使うと、NPCの作成が非常に簡単！

もちろんデメリットもある。
Bakeがが想定通りの結果にならなかった場合の、
微調整が困難らしい・・・。

それでも利用シーンはたくさんあるし、
有効に使っていきたい機能の一つだと思う。

■参考サイト

Unity公式マニュアル
https://docs.unity3d.com/ja/current/Manual/Navigation.html

テラシュールブログ
http://tsubakit1.hateblo.jp/entry/20131104/1383573538

ウダサンコウボウ
http://udasankoubou.blogspot.jp/2014/01/unity_28.html

ご清聴ありがとうございました